

### CLAIMS

1. In a graphics environment including a graphics application communicating with a graphics system through a first driver interfaced directly with the graphics application and a second driver interfaced directly with the graphics system, said drivers communicating with each other using a predetermined graphics interface, a system for identification and assessment of performance optimizations implemented in the graphics environment, said identification and assessment of said performance optimizations based upon an optimized graphics call sequence generated by an application of one or more optimizations applied to a captured graphics call sequence occurring between said first and second drivers.
2. The system of claim 1, comprising:  
a plurality of graphics call sequence optimizers, at least one of which processes at least a portion of said captured graphics call sequence to generate an intermediate optimized graphics call sequence.
3. The system of claim 2, wherein said plurality of graphics call sequence optimizers include at least one graphics call sequence optimizer for processing said intermediate optimized graphics call sequence to produce said optimized graphics call sequence.
4. The system of claim 1, comprising:  
a graphics call sequence optimizer for processes at least a portion of said captured graphics call sequence to generate said optimized graphics call sequence.
5. The system of claim 1, wherein said plurality of graphics call sequence optimizers comprises:  
a graphics state call coalescer constructed and arranged to eliminate from a continuous series of graphics state calls in said captured graphics call sequence contains redundant, conflicting or otherwise unnecessary graphics state calls.
6. The system of claim 1, wherein said plurality of graphics call sequence optimizers comprises:

a primitive state coalescer constructed and arranged to coalesce graphics vertex calls contained in a continuous series of primitive command sets occurring in said captured graphics state call that render primitives of the same type.

7. The system of claim 6, wherein said API is the OpenGL API, and wherein said primitive command sets comprise:

a glBegin()/glEnd() graphics call pair; and  
at least one intervening graphics vertex call.

8. The system of claim 7, wherein said primitive state coalescer removes all glBegin() graphics calls and glEnd() graphics calls other than a glBegin() graphics call occurring first in said continuous series of primitive command sets and a glEnd() graphics call occurring last in said continuous series of primitive command sets.

9. A method for optimizing a first graphics system command set having a sequence of graphics calls generated in accordance with a particular graphics application programming interface (API) to generate an optimized command set, the method comprising the steps of:

(a) capturing said sequence of graphics calls; and  
(b) restructuring said captured sequence of graphics calls to produce said optimized command set.

10. The method of claim 9, wherein said restructuring step (b) comprises a step of:

(1) state coalescing a series of graphics state calls in said captured sequence of graphics calls to generate a corresponding coalesced series of graphics state calls in said optimized command set effecting a same state change in a graphics system as said captured sequence of graphic calls.

11. The method of claim 10, wherein said state coalescing step (1) comprises a step of:

(i) determining a final state value resulting from said series of graphics state calls, said final state value being determined when a non-state graphics call appears in said captured sequence of graphics calls.

12. The method of claim 11, wherein said state coalescing step (1) further comprises a step of:

(ii) updating a current state value to said final state value when a non-state graphics call appears and when said final state value is not equal to said current state value.

5 13. The method of claim 10, wherein said state coalescing step (1) begins determining a final state value when a first graphics state call appears in said captured sequence of graphics calls and wherein said state coalescing step (1) ends when a non-state graphics call appears in said captured sequence of graphics calls.

14. The method of claim 9, wherein restructuring step (b) comprises the step of:

10 (1) primitive coalescing a series of graphics primitive calls in said captured sequence of graphics calls to generate a corresponding coalesced series of graphics primitive calls in said optimized command set effecting a same primitives rendering in a graphics system as said captured sequence of graphic calls.

15 15. The method of claim 14, wherein said primitive coalescing step (1) comprises a step of:

(i) sequentially combining a series of *like* graphics primitive calls, *if such combination is possible (i.e., cannot combine polygons)*, from said captured sequence of graphic calls until a non-primitive graphics call appears in said captured sequence of graphics calls.

20 16. The method of claim 15, wherein said primitive coalescing step (1) further comprises the steps of:

(ii) outputting said sequentially combined series of like graphics primitive calls when a non-primitive graphics call appears in said captured sequence of graphics calls;

25 (iii) outputting said sequentially combined series of like graphics primitive calls when the current graphics primitive call is not of the same type as the previous graphics primitive call; and

(iv) outputting a command to cause said graphics system to render said sequentially combined series of like primitive graphics calls when a graphics frame call appears in said captured sequence of graphics calls.

17. The method of claim 9, wherein said restructuring step (b) comprises the steps of:

(1) state coalescing a series of graphics state calls in said captured sequence of graphics calls to generate a corresponding coalesced series of graphics state calls in said optimized command set effecting a same state change in said graphics system as said

5 captured sequence of graphic calls; and

(2) primitive coalescing a series of graphics primitive calls in said captured sequence of graphics calls to generate a corresponding coalesced series of graphics primitive calls in said optimized command set effecting a same primitives rendering in said graphics system as said captured sequence of graphic calls.

10 18. The method of claim 9, wherein said restructuring step (b) comprises a step of:

(1) state grouping a plurality of graphics primitive call sequences in said captured sequence of graphics calls to generate a corresponding state grouped plurality of graphics primitive calls in said optimized command set effecting a same state change and a same primitives rendering in a graphics system as said captured sequence of graphic calls.

15 19. The method of claim 18, wherein said state grouping step (1) comprises a step of:

(i) grouping sequences of graphics primitives calls having the same graphics state calls characteristics.

20. The method of claim 19, wherein said state grouping step (1) further comprises a step of:

20 (ii) outputting a command to cause said graphics system to render said state grouped sequences of graphics primitive calls when there are no more calls in said captured sequence of graphics calls;

(iii) clearing and outputting a command to cause said graphics system to render said state grouped sequences of graphics primitive calls when a graphics frame call appears in said captured sequence of graphics calls; and

25 (iv) updating a current state value when a graphics state call appears in said captured sequence of graphics calls.